The Data Mining OPtimization Ontology

C. Maria Keet[a], Agnieszka Ławrynowicz[b], Claudia d'Amato[c], Alexandros Kalousis[d], Phong Nguyen[e], Raul Palma[f], Robert Stevens[g], Melanie Hilario[h]

[a]*Department of Computer Science, University of Cape Town, South Africa, mkeet@cs.uct.ac.za*
[b]*Institute of Computing Science, Poznan University of Technology, Poland, agnieszka.lawrynowicz@cs.put.poznan.pl*
[c]*Department of Computer Science, University of Bari, Italy, claudia.damato@uniba.it*
[d]*Department of Business Informatics, University Of Applied Sciences, Switzerland, Alexandros.Kalousis@hesge.ch*
[e]*Department of Computer Science, University of Geneva, Switzerland, Phong.Nguyen@unige.ch*
[f]*Poznan Supercomputing and Networking Center, Poland, rpalma@man.poznan.pl*
[g]*School of Computer Science, University of Manchester, United Kingdom, robert.stevens@manchester.ac.uk*
[h]*Artificial Intelligence Laboratory, University of Geneva, Switzerland, melanie.hilario@unige.ch*

Abstract

The Data Mining OPtimization Ontology (DMOP) has been developed to support informed decision-making at various choice points of the data mining process. The ontology can be used as a reference by data miners and deployed in ontology-driven information systems. The primary purpose for which DMOP has been developed is the automation of algorithm and model selection through semantic meta-mining that makes use of an ontology-based meta-analysis of complete data mining processes in view of extracting patterns associated with mining performance. To this end, DMOP contains detailed descriptions of data mining tasks (e.g., learning, feature selection), data, algorithms, hypotheses such as mined models or patterns, and workflows. A development methodology was used for DMOP, including items such as competency questions and foundational ontology reuse. Several non-trivial modeling problems were encountered and due to the complexity of the data mining details, the ontology requires the use of the OWL 2 DL profile. DMOP was successfully evaluated for semantic meta-mining and used in constructing the Intelligent Discovery Assistant, deployed at the popular data mining environment RapidMiner.

Keywords: Ontology, OWL, data mining, meta-learning, semantic meta-mining

1. Introduction

The primary goal of the Data Mining OPtimization Ontology (DMOP, pronounced dee-mope) is to support all decision-making steps that determine the outcome of the data mining (DM) process. The DM process is standardized by CRISP-DM [1], a high-level standard DM process model. According to CRISP-DM, the DM process is composed of the following phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The optimization of a DM process requires knowledge of how components from its different phases interact and how their internal characteristics influence its performance. Despite the existence of CRISP-DM, the optimization of the DM process was not possible because the necessary additional detailed knowledge was missing. This means that a comprehensive analysis of DM processes was not possible, and, consequently nor was the *optimization* of the performance of the DM process. DMOP fills this gap. It can be used by data mining practitioners to inform manual selection of various ingredients (algorithms, models, and parameters) used for constructing DM processes. Most of all, however, DMOP has been designed to *support the automation of such selections* with a novel form of meta-learning, named *semantic meta-mining* [2].

Meta-learning [3], or learning to learn, is defined in computer science as the application of machine learning techniques to meta-data about past machine learning experiments with the goal of modifying some aspects of the learning process in order to improve the performance of the resulting model.

Traditional meta-learning focused only on the central (modeling) phase of the DM process, where machine learning algorithms are executed to build a model. However, the quality of the mined model depends strongly also on other phases of a DM process. Traditional meta-learning regarded learning algorithms as black boxes, correlating the observed performance of their output (learned model) with characteristics of their input (data). However, the algorithms that have the same types of input/output may differ in internal characteristics.

Semantic meta-mining is distinguished from traditional meta-learning by the following three properties. First, it extends the meta-learning approach to *meta-mining*, i.e. learning from the full DM process. Secondly, it is co-driven by knowledge of DM process and its components represented in the DM ontology and knowledge base (KB), in contrast to purely data driven traditional meta-learning. Thirdly, it breaks open the black box by explicitly analyzing DM algorithms along various dimensions to correlate observed performance of learned hypotheses resulting from DM processes with both data and algorithm characteristics. Semantic meta-mining is thus an ontology-based, process-oriented form of meta-learning that exploits in-depth knowledge of DM processes. To support se-

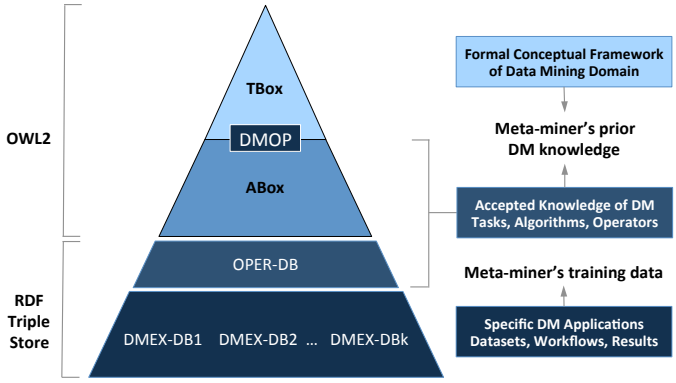

Figure 1: Architecture of the ontology, its associated knowledge base, operator database, and satellite triple stores.

mantic meta-mining, DMOP contains a detailed taxonomy of algorithms used in DM processes, each described in terms of its underlying assumptions, the cost functions and adopted optimization strategies, the classes of hypotheses (models or pattern sets) it generates, and other properties. Following such a "glass box" approach makes explicit internal algorithm characteristics. This allows meta-learners using DMOP to generalize over algorithms and their properties, including those algorithms that did not appear in the training set.

Performing semantic meta-mining requires knowledge about various layers of data mining experiments, which are reflected in the DMOP architecture (see Fig. 1): a top-layer with the formal conceptual framework of the data mining domain (e.g. algorithm class specification), a middle layer of accepted knowledge about the DM domain (e.g. particular algorithms and their known implementations), and a bottom layer of application specific DM data (e.g. datasets, workflows, results).

The two-tiered top layer in the figure represents DMOP (denoted TBox) and its KB (denoted ABox), where the latter uses knowledge from DMOP to model existing data mining algorithms. Both, DMOP and its associated KB, are implemented in OWL 2. An RDF database (OPER-DB) contains descriptions of operators, i.e., implementations of algorithms described in DMOP and particularly those implementations that are a part of popular DM software (such as RapidMiner or Weka). The ABox, together with an operator database, provides accepted knowledge about DM tasks, algorithms and operators. Altogether these are application-independent resources that constitute the meta-miner's prior DM knowledge. Meta-data recorded during data mining experiments are described using DMOP and its associated resources, and thus constitute application-specific training and testing data for a meta-miner. They are stored in application-dedicated RDF triple stores (denoted DMEX-DBs in the figure) and describe datasets, workflow descriptions, and data mining experiments.

This paper describes v5.4 of DMOP that can be downloaded from `http://www.dmo-foundry.org`. The ontology in this version has 758 classes, 169 object properties, 15 data properties and 3202 axioms.

DMOP provides a unified conceptual framework for analyzing DM tasks, algorithms, models, datasets, workflows and performance metrics, and their relationships, as described in Sect. 3 whilst methodological aspects are described in Sect. 2. To fulfill requirements of this in-depth analysis, we have encountered a number of non-trivial modeling issues in DMOP development, of which the main ones are discussed in Sect. 4. DMOP's goals and required coverage resulted in using almost all OWL 2 features. DMOP was successfully applied in semantic meta-mining, and deployed in RapidMiner data mining environment (download statistics provided) as described in Sect. 5. Conclusions are drawn in Sect. 6.

2. Ontology development

There are several methodologies for ontology development in the literature, including METHONTOLOGY [4], NeON [5], Melting Point [6], and DiDOn [7]. Although these methodologies may differ in scope and focus, they have some commonalities that can be roughly mapped in three main stages of the ontology development process: 1) specification with a domain analysis (including use cases, competency questions), 2) the conceptualization, formalization, and implementation, and 3) maintenance with refinement and evolution of the ontology. We did not use one specific methodology, but took usable components from the extant methodologies and tailored it to the specific case at hand.

During the first stage, requirements with competency questions—i.e., questions that an ontology should be able to answer—were formulated (see Sect. 2.1), the use cases for semantic meta-mining specified, such as providing DM expertise to an intelligent knowledge discovery assistant (see Sect. 5.1), and related domain ontologies were investigated and assessed to what extent they would be able to meet the requirements (see Sect. 2.2). The outcome of that stage fed into stage two, leading to a design of the DMOP architecture, and the subsequent ontology authoring by people residing at different institutions and with overlapping and complementary knowledge of the subject domain. The tool used was Protégé 4.x[1]. The ontology has gone through various cycles of design and evaluation, including a testing phase on meeting the requirements. Besides content experts, also ontology experts were consulted, who provided additional modeling guidance and solutions. The ontology is in the third stage since late 2011, where novel methods and tools at the level of axiom enhancement and 'debugging' are being used, such as [8, 9], new sections have been added, such as on clustering, and, as the ontology became larger and more complex, more structure has been added to the ontology by aligning it to a foundational ontology (see Sect. 2.3), which are the main changes that resulted into a v5.3 and v5.4 of DMOP.

There are now three ways of contributing to the ontology, each targeted to a different type of contributor. Mode 1 is the open, bottom-up collaborative ontology development approach for domain and/or ontology experts, which relies on Cicero Argumentation Tool[2] and the DMOP forum for input and the Editorial Board to review community input.

[1] `http://protege.stanford.edu`
[2] `http://cicero.uni-koblenz.de/wiki/index.php/Download`

Mode 2: Data miners not familiar with ontology tools can fill in predesigned templates—alike a user-friendly Ontology Design Content Pattern—to populate areas of the ontology with relatively stable concept and property definitions, e.g. relating operators to their algorithms, which will be screened by the ontology's Editorial Board prior to integration into the target ontology. Mode 3: The contributor is a data mining expert and conversant with ontology development who not only contributes new data mining content, but also defines new concepts and relations needed for content formalization, so that the domain expert on her specific topic will impact ontology design (at least locally) and conceptualization. The expert contributor will develop the assigned module using her preferred ontology editor, and will submit it to the ontology's Editorial Board in the form of an OWL file. After validation, the module will become an integral part of the ontology. These modes of collaboration are accessible from `http://www.dmo-foundry.org`.

We will highlight three salient aspects of the process followed: the *competency questions*—important for the success of deployment of the ontology; *related domain ontologies* to assess to what extent we could reuse existing domain ontologies in data mining; and the *alignment of DMOP with a foundational ontology*.

2.1. DMOP Competency questions

The principal competency question for the DMOP was:

CQ1.1 Given a data mining task/data set, which of the valid or applicable workflows/algorithms will yield optimal results (or at least better results than the others)?

This competency question is decomposed into many other questions and we present a selection of them here. Coarse-grained questions include:

CQ2.1 Given a set of candidate workflows/algorithms for a given task/data set, which data set characteristics should be taken into account in order to select the most appropriate one?

CQ2.2 Given a set of candidate workflows/algorithms for a task/data set, which workflow/algorithm characteristics should be taken into account in order to select the most appropriate one?

which can be refined into more detailed questions, such as:

CQ3.1 Are there learning algorithms that I can use on high-dimensional data without having to go through preliminary dimensionality reduction?

CQ3.2 Which induction algorithms should I use (or avoid) when my dataset has many more variables than instances?

CQ3.3 Which learning algorithms perform best on microarray or mass spectrometry data?

How these are satisfied will be discussed in Sect. 3.5 and 5.2.

2.2. Related domain ontologies

An overview of early approaches to methodical descriptions of DM processes may be found in [2]. The majority of work concerning formal representation of data mining in ontology languages is aimed at the construction of DM workflows. One strand of this research deals with the development of distributed DM applications on the Grid [10, 11]. The pre-OWL DAMON ontology provides a characterization of available data mining software in order to enable semantic searching for appropriate DM resources and tools [10]. The ontology of GridMiner Assistant (GMA) [11] aims to support dynamic, interactive construction of DM workflows in Grid-enabled data mining systems.

Other ontologies developed for DM workflow construction are KDDONTO [12], KD ontology [13] and DMWF [14], all of them using OWL as a major representation language. These ontologies focus on modeling an algorithms' inputs/outputs to enable generation of valid compositions of them. For instance, a Hierarchical Task Network (HTN) based planner eProPlan [14], uses DMWF to plan a set of valid workflows based on operator (algorithm implementation) preconditions and effects modeled in DMWF by means of SWRL[3] rules.

Few existing DM ontologies go beyond supporting workflow construction. OntoDM [15] aims to provide a unified framework for data mining and contains definitions of the basic data mining concepts, but lacks a particular use case. Exposé [16] aims to provide a formal domain model for a database of data mining experiments. It uses OntoDM together with the data mining algorithms from DMOP, and a description of experiments (algorithm setup, execution, evaluation) to provide the basis of an experiment markup language. The primary use of OntoDM and Exposé may thus be viewed as providing controlled vocabulary for DM investigations.

None of the related ontologies was developed with the goal of the optimization of the performance of DM processes, what is expressed by our principal competency question. They do not provide sufficient level of details needed to support semantic meta-mining. In particular, the ontologies that are focused on workflow construction do not model the internal characteristics of algorithms (cf. competency question CQ2.2) but just their inputs and outputs. Hence they help in answering the question how to build a valid workflow, but not necessarily how to build an *optimal* workflow.

2.3. Alignment of DMOP with a foundational ontology

There are multiple good reasons to use a foundational ontology in theory, and it has been shown to improve the ontology quality, understandability, and interoperability in praxis [17]. It comes at the 'cost' for figuring out how to align a domain ontology with it, and it can have implications for the language used for the overall ontology. The principal issues from a language viewpoint are: 1) to import or to extend, 2) if import, whether that should be done in whole or just the relevant module extracted from the foundational ontology, 3) how to handle the differences in expressiveness that may exist—and possibly be required—between the foundational ontology and the domain ontology, and 4) how to rhyme different modeling 'philosophies' between what comes from Ontology, what is represented in foundational ontologies, and what is permitted in OWL (i.e.,

[3] `http://www.w3.org/Submission/SWRL`

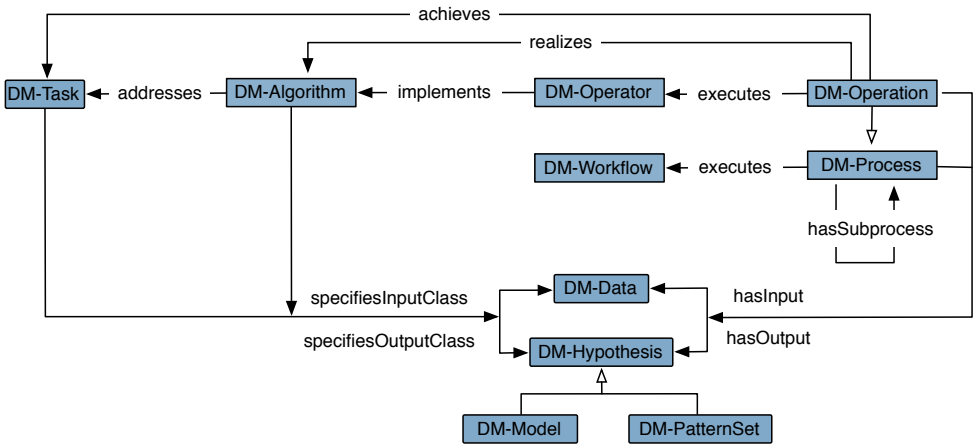

Figure 2: Simplified overview of the core concepts of DMOP.

features that are objectionable from an ontological viewpoint, such as class-as-instance, nominals, and data properties).

The two main reasons to align DMOP with a foundational ontology were the modeling issue about attributes and data properties anyway, where extant non-foundational ontology solutions were partial re-inventions of how they are treated in a foundational ontology (see Sect. 4.3), and the reuse of the ontology's object properties. DOLCE, GFO, and YAMATO are available in OWL and have extensive entities on 'attributes' and many reusable object properties. Both a manual assessment and an automated recommender (ONSET v1.2 [18]) were used to determine the comparatively 'optimal' foundational ontology for DMOP given its requirements, which was DOLCE, therefore we will only summarize how DMOP was mapped to DOLCE [19]. Determining the suitable DOLCE category for alignment and carrying out the actual mapping has been done manually; some automation to suggest mappings would be a welcome addition.

3. DMOP's contents

The core concepts of DMOP (Fig. 2) are the different ingredients that go into the data mining process (DM-Process):

- The input of the process is composed of a task specification (DM-Task) and training/test data (DM-Data) provided by the user;
- Its output is a hypothesis (DM-Hypothesis), which can take the form of a global model (DM-Model) or a set of local patterns (DM-PatternSet).

Tasks and algorithms are not processes that directly manipulate data or models, rather they are specifications of such processes:

- A DM-Task specifies a DM process (or any part thereof) in terms of the input it requires and the output it is expected to produce.
- A DM-Algorithm is the specification of a procedure that addresses a given DM-Task, while a DM-Operator is a program that implements a given DM-Algorithm and that is executed by a DM-Operation.
- Instances of DM-Task and DM-Algorithm do no more than *specifying* their input/output types (only processes have actual inputs and outputs).

Some of the object properties of DM processes are:

- It hasInput and it hasOutput some IO-Object (DM-Data or DM-Hypothesis);
- A process that executes a DM operator also realizes the DM algorithm that isImplementedBy that operator;
- A DM algorithm addresses a DM task, and the process achieves the DM task addressed by the algorithm.

Finally, a DM-Workflow is a complex structure composed of DM-operators, and a DM-Experiment is a complex process composed of operations (or operator executions). An experiment is described by all the objects that participate in the process: a workflow, data sets used and produced by the different data processing phases, the resulting models, and meta-data quantifying their performance.

3.1. DM Tasks

The top-level DM tasks listed below are defined by their inputs and outputs.

A DataProcessingTask receives and outputs data. Its four subclasses produce new data by cleansing (DataCleaningTask), reducing (DataReductionTask), extracting a compact representation (DataAbstractionTask) or otherwise transforming the input data (DataTransformationTask). These classes are further articulated in subclasses representing more fine-grained tasks.

An InductionTask consumes data and produces hypotheses. It can be either a ModelingTask or a PatternDiscoveryTask, based on whether it generates hypotheses in the form of global models or local pattern sets. Modeling tasks can be predictive (e.g. classification) or descriptive (e.g., clustering), while pattern discovery tasks are further subdivided into classes based on the nature of the extracted patterns: associations, dissociations, deviations, or subgroups.

A HypothesisProcessingTask consumes hypotheses and transforms (e.g., rewrites or prunes) them to produce enhanced—less complex or more readable—versions of the input hypotheses. A HypothesisEvaluationTask quantifies the quality of an induced

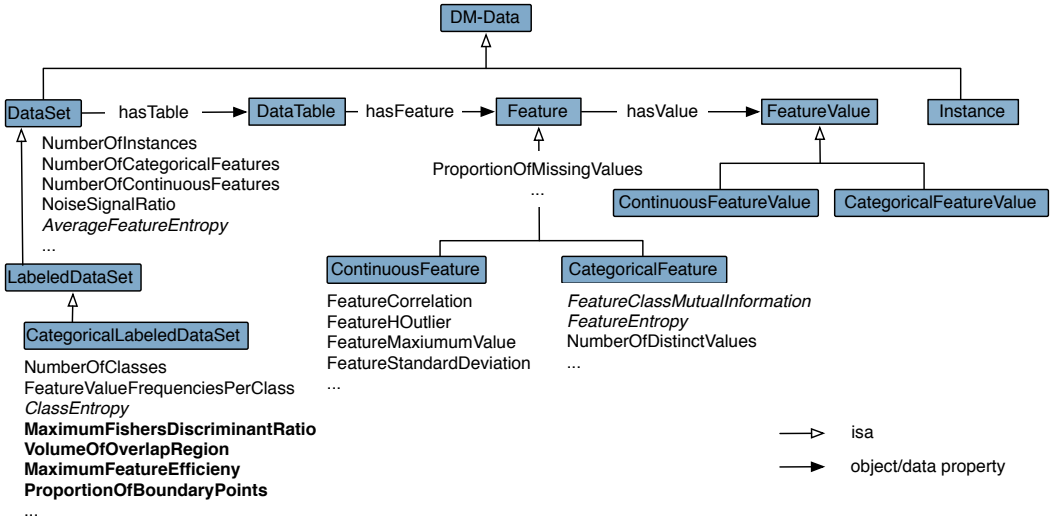

Figure 3: Data characteristics modeled in DMOP. Rectangles: subclasses of DM-Data class; unbounded text near the rectangles denote subclasses of the DataCharacteristic class associated to a DM-Data class through an OWL object property, where those in italics font are information-theoretic measures and the ones in bold are geometric indicators.

hypothesis with respect to a specific criterion (e.g., predictive performance). A HypothesisApplicationTask applies an induced hypothesis to new data.

3.2. Data

As the primary resource that feeds the knowledge discovery process, data have been a natural research focus for data miners. Over the past decades meta-learning researchers have actively investigated data characteristics that might explain generalization success or failure. Fig. 3 shows the characteristics associated with the different Data subclasses (shaded boxes). Most of these are statistical measures, such as the number of instances or the number of features of a data set. Others are information-theoretic measures (italicized in the figure). Characteristics in bold font are geometric indicators of data set complexity, such as the maximum value of Fisher's Discriminant Ratio that measures the highest discriminatory power of any single feature in the data set (see [20] for detailed definitions).

3.3. DM Algorithms

The top levels of the DM-Algorithm hierarchy reflect those of the DM-Task hierarchy, since each algorithm class is defined by the task it addresses. However, the DM-Algorithm hierarchy is much deeper than the DM-Task hierarchy: for each leaf of the task hierarchy, there is often a dense subhierarchy of algorithms that specify diverse ways of addressing each task. For instance, the leaf concept ClassificationModelingTask maps directly onto the ClassificationModelingAlgorithm class, whose three main subclasses [21] are illustrated in the following. A GenerativeAlgorithm computes the class-conditional densities $p(\mathbf{x}|C_k)$ and the priors $p(C_k)$ for each class C_k. Examples of generative methods are normal (linear or quadratic) discriminant analysis and Naive Bayes. A DiscriminativeAlgorithm, such as logistic regression, computes posterior probabilities $p(C_k|\mathbf{x})$ directly to determine class membership. A DiscriminantFunctionAlgorithm builds a direct mapping $f(\mathbf{x})$ from input $\mathbf{x}$ onto a class label; neural networks and support vector classifiers (SVCs) are examples of discriminant function methods. These three DM-Algorithm families spawn multiple levels of descendant classes that are distinguished by the type and structure of the models they generate.

One innovative feature of DMOP is the modeling and exploitation of algorithm properties in meta-mining. All previous research in meta-learning has focused exclusively on data characteristics and treated algorithms as black boxes. DMOP-based meta-mining brings to bear in-depth knowledge of algorithms as expressed in their elaborate network of object properties. One of these is the object property has-quality, which relates a DM-Algorithm to an AlgorithmCharacteristic (Fig. 4). A few characteristics are common to all DM algorithms; examples are characteristics that specify whether an algorithm makes use of a random component, or handles categorical or continuous features. Most other characteristics are subclass-specific. For instance, characteristics such as LearningPolicy (Eager/Lazy) are common to induction algorithms in general, whereas Tolerance-ToClassImbalance and HandlingOfClassificationCosts make sense only for classification algorithms.

Note that has-quality is only one among the many object properties that are used to model DM algorithms. An induction algorithm, for instance, requires other properties to fully model its inductive bias. Some examples are the properties: assumes which expresses its underlying assumptions concerning the training data; specifiesOutputClass which links to the class of models generated by the algorithm, making explicit its hypothesis language or representational bias; hasOptimizationProblem which identifies its optimization problem and the strategies followed to solve it, thus defining its preference or search bias.

3.4. Content alignment to DOLCE

The following subsumption axioms were added to align DMOP with DOLCE. DOLCE's dolce:process in the perdu-

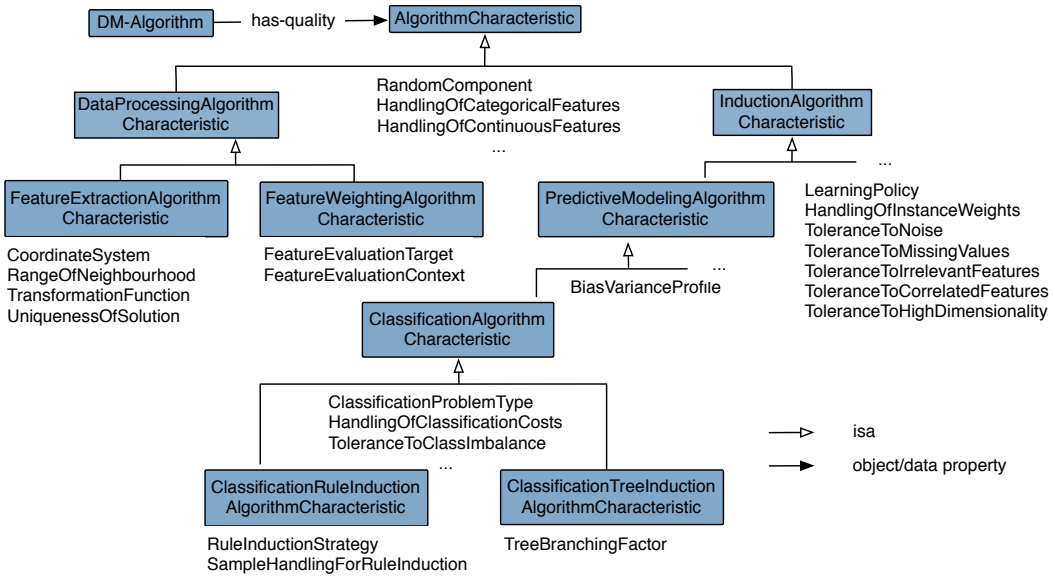

Figure 4: Data mining algorithm characteristics: the main top-level classes and a selection of their attributes (subclasses of Characteristic).

rant branch now has as subclasses DM-Experiment and DM-Operation, whereas most DM classes, such as algorithm, software, strategy, task, and optimization problem, are subclasses of dolce:non-physical-endurant. Characteristics and parameters of such entities have been made subclasses of dolce:abstract-quality, and for identifying discrete values, classes were added as subclasses of dolce:abstract-region. That is, each of the four DOLCE main branches have been used. Regarding object properties, DMOP reuses mainly DOLCE's parthood, quality, and quale relations. Mapping DMOP into DOLCE had the most effect regarding representing DM characteristics and parameters ('attributes'), which is discussed in Sect. 4.3.

3.5. Answering competency questions

The competency questions may be divided into two groups: those that may be already answered by the DMOP's KB and those that may be answered with use of the DMOP based meta-mined model, the product of semantic meta-mining. The latter ones that are related to performance of DM processes will be discussed in Sect. 5. The former ones are the questions that deal with characteristics of particular DM entities. For instance, the competency question CQ3.1 may be answered by querying for the DM algorithms whose characteristic ToleratesHighDimensionality has quale 'Yes', that is those that have the characteristic of tolerating high dimensionality. In the DL Query notation of Protégé this query would be formulated as "DM-Algorithm and has-quality value ToleratesHighDimensionality". The retrieved results consist of the algorithm families (classes ClassificationRuleInductionAlgorithm, ClassificationTreeInductionAlgorithm, and SVC-Algorithm) and particular algorithms (e.g. members C4.5, C4.5Prob, CARTc, CHAID, DecisionStump, ID3, LogisticModelTree, NBTree, RandomTree of the ClassificationTreeInductionAlgorithm class).

4. Modeling challenges

In this section we present the main modeling choices, issues arisen, and solutions adopted, therewith providing some background as to why certain aspects from the overview in the preceding section are modeled the way they are.

4.1. Meta-modeling in DMOP

Right from the start of DMOP development, one of the most important modeling issues concerning DM algorithms was to decide whether to model them as classes or individuals. Though DM algorithms may have different implementations, the common view is to see particular algorithms as single instances, and not collections of instances. However, the modeling problem arises when we want to express the types of inputs and outputs associated with a particular algorithm. We will describe this problem and how it was solved by means of an example, illustrated in Fig. 5.

Recall that: i) only processes (executions of workflows) and operations (executions of operators) consume inputs and produce outputs; ii) DM algorithms (as well as operators and workflows) can, in turn, only specify the type of input or output; iii) inputs and outputs (DM-Dataset and DM-Hypothesis class hierarchy, respectively) are modeled as subclasses of IO-Object class. Then expressing a sentence like "the algorithm C4.5 specifiesInputClass CategoricalLabeledDataSet" became problematic. Based on our original design (reflected in Fig. 5a), it would mean that a particular algorithm (C4.5, an instance of the DM-Algorithm class) specifies a particular type of input (CategoricalLabeledDataSet, a subclass of DM-Hypothesis class), but classes cannot be assigned as property values to individuals in OWL.

Our initial solution to tackle this problem was creating one artificial class per each single algorithm with a single instance corresponding to this particular algorithm, as recommended in

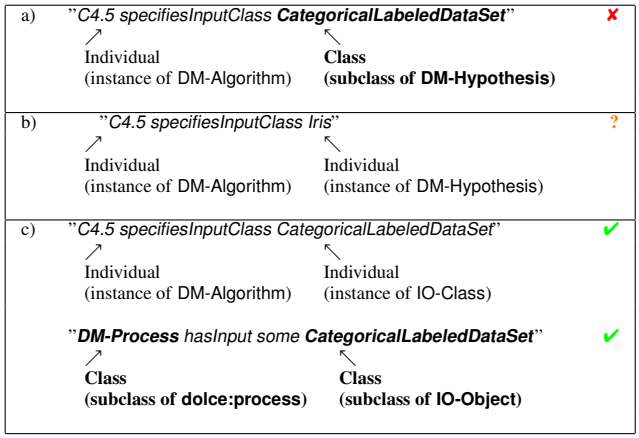

Figure 5: Illustration of a modeling problem and its solution based on metamodeling. a) Original design problem: expressing types of inputs/outputs associated with an algorithm; b) Initial solution: one artificial class per each single algorithm with a single instance corresponding to this particular algorithm; c) Final solution: weak form of punning available in OWL 2; IO-Class as meta-class of all classes of input and output objects.

[22] (e.g. C4.5Algorithm class with single instance C4.5). However, in our case, such modeling led to technical problems. Since each of the four properties—hasInput, hasOutput, specifiesInputClass, specifiesOutputClass—were assigned a common range—IO-Object—it opened a way to make problematic ABox assertions like C4.5 specifiesInputClass Iris, where Iris is a concrete dataset. Clearly, any DM algorithm is not designed to handle *only* a particular dataset.

In our final solution, we decided to use the weak form of punning available in OWL 2 (see Fig. 5c). We had noticed that CategoricalLabeledDataSet could be perceived as an instance of a meta-class—the class of all classes of input and output objects, named IO-Class in DMOP. In this way, the sentence C4.5 specifiesInputClass CategoricalLabeledDataSet delivered the intended semantics. However, we also wanted to express sentences like DM-Process hasInput some CategoricalLabeledDataSet. The use of the same IO object (like CategoricalLabeledDataSet) once as a class (subclass of IO-Object) and at other times as an instance required some form of meta-modeling. In order to implement it, we investigated some available options. This included an approach based on an axiomatization of class reification proposed in [23], where in a metamodeling-enabled version $\mathcal{O}^{meta}$ of a given ontology $\mathcal{O}$, class-level expressions from $\mathcal{O}$ are transformed into individual assertions such that each model of $\mathcal{O}^{meta}$ has two kinds of individuals, those representing classes and those representing proper individuals, and meta-level rules are encoded in class level. We chose not to follow this technique due to its possible efficiency issues.

Punning in our approach is only applied to leaf-level classes of IO-Object; non-leaf classes are not punned but represented by associated meta-classes, e.g., the IO-Object subclass DataSet maps to the IO-Class subclass DataSetClass. Similarly, the instances of DM-Hypothesis class represent individual hypotheses generated by running an algorithm on the particular dataset, while the class DM-HypothesisClass is the meta-class whose instances are the leaf-level descendant classes of DM-Hypothesis. Except for the leaf-level classes, the IO-Class hierarchy structure mimics that of the IO-Object hierarchy.

4.2. Property chains in DMOP

DMOP has 11 property chains, which have been investigated in detail in [8]. The principal issues in declaring safe property chains, i.e., that are guaranteed not to cause unsatisfiable classes or other undesirable deductions, are declaring and choosing properties, and their domain and range axioms. To illustrate one of the issues in declaring property chains, we use hasMainTable $\circ$ hasFeature $\sqsubseteq$ hasFeature: chaining requires compatible domains and ranges at the chaining 'points', such as the range of hasMainTable and domain of hasFeature, and with the domain and range of the property on the right-hand side. In this case, hasFeature's domain is DataTable that is a sister-class of hasMainTable's domain DataSet, but the chain forces that each participating entity in hasFeature has to be a subclass of its declared domain class, hence DataSet $\sqsubseteq$ DataTable is derived to keep the ontology consistent.

Ontologically, this is clearly wrong, and hasFeature's domain is now set to DataSet or DataTable. Each chain has been analysed in a similar fashion and adjusted where deemed necessary (see [8] for the generic set of tests and how to correct any flaws for any property chain).

DMOP contains more elaborate property chains than the aforementioned one. For instance, realizes $\circ$ addresses $\sqsubseteq$ achieves, so that if a DM-Operation realizes a DM-Algorithm that addresses a DM-Task, then the DM-Operation achieves that DM-Task, and with the chain implements $\circ$ specifiesInputClass $\sqsubseteq$ specifiesInputClass, we obtain that when a DM-Operator or OperatorParameter implements an AlgorithmParameter or DM-Algorithm and that specifies the input class IO-Class, then the DM-Operator or OperatorParameter specifies the input class IO-Class.

4.3. Qualities and attributes

A seemingly straightforward but actually rather intricate, and essentially unresolved, issue is how to handle 'attributes' and, in a broader context, measurements in OWL ontologies. For instance, each FeatureExtractionAlgorithm has as an 'attribute' a transformation function that is either linear or non-linear. One might be tempted to take the easy way out and simply reuse the "UML approach" where an attribute is a binary relation between a class and a datatype; e.g., with a simplified non-DMOP intuitive generic example, given a data property hasWeight with as XML data type `integer`, one can declare Elephant $\sqsubseteq$ =1 hasWeight.`integer`. And perhaps a hasWeightPrecise with as data type `real` may be needed elsewhere. And then it appears later on that the former two were assumed to have been measured in kg, but someone else using the ontology wants to have it in lbs, so we would need another hasWeightImperial, and so on. Essentially, with this approach, we end up with exactly the same issues as in database integration, precisely what ontologies were supposed to solve. Instead of building into one's ontology application decisions about how to store the data in the information system (and in which unit it is), one can generalize the (binary) attribute into a class, reuse the very notion

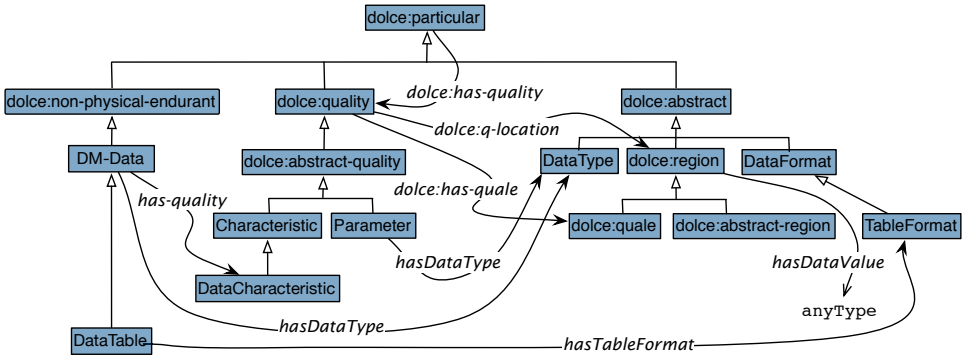

Figure 6: Condensed section and partial representation of DMOP regarding 'attributes'.

of Weight that is the same in all cases, and then have different relations to both value regions and units of measurement. This means unfolding the notion of an object's property, like its weight, from one attribute/OWL data property into at least two properties: one OWL object property from the object to the 'reified attribute'—a so-called "quality property", represented as an OWL class—and then another property to the value(s). The latter, more elaborate, approach is favored in foundational ontologies, especially in DOLCE, GFO and YAMATO. DOLCE uses the combination Endurant that has a qt relation to Quality (disjoint branches) that, in turn, has a ql relation to a Region (a subclass of the yet again disjoint Abstract branch). While this solves the problem of non-reusability of the 'attribute' and prevents duplication of data properties, neither ontology has any solution to representing the actual values and units of measurements. But they are needed for DMOP too, as well as complex data types, such as an ordered tree and a multivariate series.

We considered related work on qualities, measurements and similar proposals from foundational ontologies, to general ontologies, to domain ontologies for the experimental sciences [19, 24, 25, 26, 27]. This revealed that the measurements for DMOP are not measurements in the sense of recording the actual measurements, their instruments, and systems of units of measurements, but more alike values for parameters, e.g., that the TreeDepth has a certain value and a LearningPolicy is eager or lazy, and that some proposals, such as OBOE [25], are versions of DOLCE's approaches[4].

This being the case, we opted for the somewhat elaborate representation of DOLCE, and added a minor extension to that for our OWL ontology in two ways (see Fig. 6): i) DM-Data is associated with a primitive or structured DataType (which is a class in the TBox) through the object property hasDataType, and ii) the data property hasDataValue relates DOLCE's Region with any data type permitted by OWL, i.e., anyType. In this way, one obtains a 'chain' from the endurant/perdurant through the dolce:has-quality property to the quality, that goes on through the dolce:q-location/dolce:has-quale property to region and on with the hasDataValue data property to the built-in data type (instead of one single data property between the endurant and the data type). For instance, we have ModelingAlgorithm $\sqsubseteq$ =1 has-quality.LearningPolicy, where LearningPolicy is a dolce:quality, and then LearningPolicy $\sqsubseteq$ =1 has-quale.Eager-Lazy, where Eager-Lazy is a subclass of dolce:abstract-region (that is a subclass of dolce:region), and, finally, Eager-Lazy $\sqsubseteq \leq 1$ has-DataValue.anyType, so that one can record the value of the learning policy of a modeling algorithm. In this way, the ontology can be linked to many different applications, who may even use different data types, yet still agree on the meaning of the characteristics and parameters ('attributes') of the algorithms, tasks, and other DM endurants.

A substantial number of classes have been represented in this way: dolce:region's subclass dolce:abstract-region has 44 DMOP subclasses, which represent ways of carving out discrete value regions for the characteristics and parameters of the endurants DM-Data, DM-Algorithm, and DM-Hypothesis. Characteristic and Parameter are direct subclasses of dolce:abstract-quality, which have 110 and 46 subclasses, respectively.

5. Usage of DMOP in semantic meta-mining

Today's DM platforms offer many algorithm implementations (operators) that support different steps of the DM process. For instance, RapidMiner (version 5.3, Community Edition) offers 688 operators, either implemented by developers of RapidMiner or acquired through the implementation of wrappers for popular DM libraries such as Weka[5]. The user of the platform must select the appropriate operators, and their combination to build a DM workflow best addressing her goal. To assist the user in the design of an effective workflow, Intelligent Discovery Assistants (IDAs) have been proposed (a recent survey of IDAs is presented in [28]). In the following, we describe how DMOP was used to construct the e-LICO IDA[6] that is the first IDA capable of both planning and ranking DM workflows. We

[4]DOLCE materials differ slightly, with quale as relation in [19] and as unary in [24] and in the DOLCE-lite.owl, and Region is a combination of a (data) value + measurement unit condensed into one (e.g. "80 kg") in [19] to deal with attribute values/qualia (there were no examples in [24] and the DOLCE-lite.owl)

[5]http://www.cs.waikato.ac.nz/ml/weka/

[6]http://www.e-lico.eu

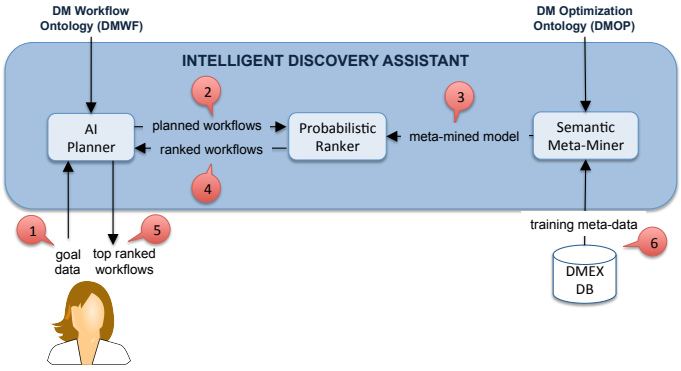

Figure 7: Intelligent Discovery Assistant; it is composed of the AI-planner, Probabilistic Ranker and Semantic Meta-Miner. The user provides the data and specifies the data mining goal (1). The AI-planner generates a (possibly huge) set of valid workflows (2). The Probabilistic Ranker ranks the workflows (4) based on the meta-mined model (3) previously computed off-line (6) by the Semantic Meta-Miner. Top ranked workflows are presented to the user (5).

discuss the evaluation of DMOP-based semantic meta-mining and the deployment of the e-LICO IDA in RapidMiner.

5.1. The e-LICO Intelligent Discovery Assistant

The e-LICO IDA has the architecture of the so-called planning-based data analysis system [29, 28] since it uses artificial intelligence (AI) planning to construct a set of workflows. The planned workflows are all valid for the given task, but there may be many of them, even billions. Therefore, the planner-based IDA exploits the results of semantic meta-mining to rank the workflows before they are presented to the user.

The architecture of the IDA is shown in Fig. 7. The user who interacts with the IDA is required to do no more than to upload annotated data (specifying roles and the types of the attributes) and to select the DM goal to be achieved (e.g. prediction) (1).

Data characteristics together with the DM Workflow Ontology (DMWF) are used by the IDA's AI-planner to generate a set of valid DM workflows (2). Valid workflows are those that fulfill the user goal, take the dataset characteristic into account, and combine operators in the way that all their pre-conditions and post-conditions are met.

Those workflows are passed to the probabilistic ranker that applies a default rule or a meta-mined model (3) computed by the semantic meta-miner to rank the workflows (4) which enables the AI planner to provide a list of top-ranked workflows to the user (5). The workflows are ranked according to the estimated values of the performance measure of the DM hypotheses they produce (for instance, for a workflow addressing the classification task, accuracy can be such a measure). Best workflows, from the functional point of view, are those that achieve relatively best values for the measure.

The meta-mined model is computed off-line by the meta-miner, which is trained on a semantic repository of meta-data of data mining experiments (DMEX-DB) based on DMOP (6).

5.2. Evaluation of DMOP-based semantic meta-mining

The meta-mined model is induced from a DMEX-DB repository that stores meta-data concerning all aspects of past DM experiments such as the dataset description, the workflow, the learned model, predictions, and performance results. The model generalizes this knowledge with use of patterns extracted from meta-data of the collection of DM workflows; DM workflows are described in terms of the presence or absence of the extracted patterns. The extracted patterns capture (structural) characteristics of the workflows and the characteristics of the workflow components. The model employs patterns to discriminate between configurations of dataset and workflow/algorithm/operator characteristics associated with good or bad performance (cf. CQ2.x). The efficacy of the model in making predictions depends on the discriminatory power of the characteristics used to induce it. The quality of the characteristics represented in DMOP is thus crucial for the meta-mined model's efficacy.

The efficacy of meta-mined models that exploit DMOP has been evaluated empirically in the following problems: predicting whether a workflow is good or bad (in terms of performance) and planning good workflows.

5.2.1. Predicting the performance of DM workflows

Building a classifier that predicts whether a workflow is in the class of the best performing workflows or in the class of the rest of the workflows was addressed by [2] and [30].

The authors of [2] evaluated two scenarios. In the first scenario, meta-mined models exploited only data characteristics. In the second scenario, meta-mined models exploited data characteristics and patterns mined from parse trees of the DM-workflow. The parse trees, that represented the order of execution of the workflow operators and their hierarchical relation, were augmented using terms from DMOP in order to derive frequent patterns over DMOP-based generalizations of the workflow components.

The experiments were conducted on the meta-data of 2275 DM experiments performed on 65 high-dimensional datasets concerning microarray experiments on different types of cancer; the datasets had many more variables than instances. The default rule (baseline) simply predicted the majority class and had 45.38 error rate. In the two semantic meta-mining scenarios, the models that were built using data and workflow characteristics performed better (38.24 error rate) than those based on data characteristics alone (40.44 error rate), and meta-mined workflow patterns proved to be discriminatory even for new algorithms and workflows (that is those not yet encountered in previous DM experiments) [2].

The capability of DMOP based meta-mined models to predict the relative performance of DM workflows was confirmed in [30]. This study used 1581 RapidMiner workflows solving a predictive modeling task on 11 UCI[7] datasets with various characteristics, whose meta-data was stored in the DMEX-DB containing over 85 million of RDF triples[8]. The workflow patterns were represented as SPARQL queries using DMOP entities. McNemar's test for pairs of classifiers was performed

[7] `http://archive.ics.uci.edu/ml/datasets.html`

[8] all experimental data, datasets, and workflows, are available at `http://www.myexperiment.org/packs/421.html`

with the null hypothesis that a classifier built using dataset characteristics and a mined pattern set has the same error rate as the baseline that used dataset characteristics and only the names of the learning DM operators. The test confirmed that classifiers trained using workflow patterns performed significantly better (accuracy of 0.927) than the baseline (accuracy of 0.890). The details of the results are available at `http://knoesis.org/resources/ijswis/209.pdf`.

The experiments proved that DMOP-based semantic meta-mining was effective in answering competency questions dealing with performance of DM algorithms and/or DM workflows. Learning algorithms performing best on microarray data (CQ3.3) and the ones that should be used or avoided when an input dataset has many more variables than instances (CQ3.2) were found in patterns resulting from meta-mining experiments described in [2]. In both mentioned studies, the computed meta-mined models proved to be effective in selecting better performing workflows from among the valid ones (CQ1.1).

5.2.2. Planning well performing DM workflows

Recall that the AI Planner constructs valid DM workflows step by step by selecting applicable operators according to their pre/post-conditions. The AI Planner alone does not have the means to differentiate between operators that have equivalent conditions since it does not take the quality of the resulting workflows into account. There may be several operators that have fitting conditions at each step.

The authors of [31] experimentally evaluated the Semantic Meta-Miner in the operator selection task. The goal was to select at a given step among a set of candidate operators the best ones to build not only valid but also optimal DM workflows. The Semantic Meta-Miner used a quality function that scored a given plan by the quality of the operators that formed the plan. The quality optimized the performance measure associated with the data mining goal of the user and the input data set.

The experiments were conducted with the same set of DM workflows as in [2]. The baseline strategy was based on the popularity of the RapidMiner's DM operators. The results were statistically significantly better for the meta-mining selection approach than for the baseline (with the average performance improvement of around 6%). The meta-mining strategy was better than the baseline in selecting the best workflow for 53 datasets out of 65. The results show the validity of the approach in planning good workflows for a given learning problem.

The experimental results for the operator selection task proved that the Semantic Meta-Miner was capable to answer which of the applicable DM algorithms would yield best results given a DM task and data set (CQ1.1). These were those implemented by best scoring DM operators and DM algorithms sharing similar characteristics with them, according to DMOP.

5.3. Deployment of the Intelligent Discovery Assistant

The meta-mined model resulting from DMOP-based semantic meta-mining is used in the IDA extension of RapidMiner developed within the e-LICO project [29].

After the IDA produces the top ranked workflows and suggests them to the user, he or she can execute the chosen workflow in RapidMiner. The data mining services required to enact the workflow (data, text, image mining) are provided by RapidAnalytics[9]. RapidAnalytics also serves as a centralized data mining experiment repository for different teams collaborating on a given application domain. It stores all relevant meta-data related to the execution of the workflow.

The raw meta-data from the RapidAnalytics repository can then be parsed and organized into a semantic repository of annotated experiments (DMEX-DB) based on DMOP. In this form, the parsed meta-data can be exploited by the meta-miner that uses DMEX-DB as the system's structured long-term memory and the source of training data.

The RapidMiner IDA Extension is available in the Rapid-I marketplace[10]. As of 3 March 2014, this RapidMiner plugin has been downloaded 8751 times (including 56 times the week of 3 March 2014). It has been bookmarked 23 times, and it is among the Top Favourites listed in the Rapid-I marketplace.

The workflows generated by the IDA can also be executed in the Taverna IDA extension[11] using an instance of a RapidAnalytics server providing RapidMiner operators as web-services. Finally, the user can upload the workflow generated by the IDA to myExperiment[12], a web portal for sharing workflows and other resources. This feature is available in both RapidMiner and Taverna.

6. Conclusions

In this paper, we have presented the DMOP ontology. It provides a conceptual framework for analyzing data mining domain–DM tasks, algorithms, models, datasets, workflows, performance metrics, and their relationships–in a way that enables *optimizing* DM processes.

While modeling data mining knowledge in DMOP, we have encountered a number of non-trivial modeling issues. These include: i) the hurdle of relating instances to classes and using classes as instances (and vv.), which has been solved by exploiting the weak form of metamodeling with OWL's punning available in OWL 2; ii) finding and resolving in a systematic way the undesirable deductions caused by property chains; iii) representation of 'attributes', where its solution is ontology-driven yet merged with OWL's data property and built-in data types to foster their reuse across applications; iv) linking to a foundational ontology. In order to properly solve these issues, we have used almost all of OWL 2's features. The resulting ontology is highly axiomatized and complex in comparison to many state-of-art domain ontologies, especially those whose primary goal is to provide common vocabulary for annotation of resources.

We described the evaluation of DMOP-based semantic meta-mining in two tasks: predicting the performance of DM workflows and planning well performing DM workflows. Finally,

[9] `http://rapid-i.com/content/view/182/196/`

[10] `http://marketplace.rapid-i.com/UpdateServer/faces/product_details.xhtml?productId=rmx_ida`

[11] `http://www.taverna.org.uk`

[12] `http://www.myexperiment.org/`

we described the usage of DMOP for constructing the Intelligent Discovery Assistant deployed at the leading data mining environment RapidMiner.

The deep modeling of the DM domain in DMOP has moved forward the field of meta-learning: traditional meta-learning has been lifted to the level of semantic meta-mining; that is, to an ontology-based form of meta-learning capable of analyzing and optimizing whole DM processes.

Acknowledgements. This work was supported by the European Union within FP7 ICT project e-LICO (Grant No 231519). Agnieszka Lawrynowicz acknowledges the support from the PARENT-BRIDGE program of Foundation for Polish Science, cofinanced from European Union, Regional Development Fund (Grant No POMOST/2013-7/8). We thank all our partners and colleagues who have contributed to the development of DMOP: Huyen Do, Simon Fischer, Dragan Gamberger, Lina Al-Jadir, Simon Jupp, Petra Kralj Novak, Babak Mougouie, Anze Vavpetic, Jun Wang, Derry Wijaya, Adam Woznica.

References

[1] Shearer, C.. The CRISP-DM model: The new blueprint for data mining. Journal of Data Warehousing 2000;5(4):13–22.

[2] Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.. Ontology-based meta-mining of knowledge discovery workflows. In: Meta-Learning in Computational Intelligence; vol. 358 of *Studies in Computational Intelligence*. Springer; 2011, p. 273–315.

[3] Jankowski, N., Duch, W., Grabczewski, K., editors. Meta-Learning in Computational Intelligence; vol. 358 of *Studies in Computational Intelligence*. Springer; 2011.

[4] Fernandez, M., Gomez-Perez, A., Pazos, A., Pazos, J.. Building a chemical ontology using METHONTOLOGY and the ontology design environment. IEEE Expert: Special Issue on Uses of Ontologies 1999;January/February:37–46.

[5] Suarez-Figueroa, M.C., de Cea, G.A., Buil, C., Dellschaft, K., Fernandez-Lopez, M., Garcia, A., et al. NeOn methodology for building contextualized ontology networks. NeOn Deliverable D5.4.1; NeOn Project; 2008.

[6] Garcia, A., O'Neill, K., Garcia, L.J., Lord, P., Stevens, R., Corcho, O., et al. Developing ontologies within decentralized settings. In: Semantic e-Science. Annals of Information Systems 11. Springer; 2010, p. 99–139.

[7] Keet, C.M.. Transforming semi-structured life science diagrams into meaningful domain ontologies with DiDOn. Journal of Biomedical Informatics 2012;45:482–494.

[8] Keet, C.M.. Detecting and revising flaws in OWL object property expressions. In: 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12); vol. 7603 of *LNAI*. Springer; 2012, p. 252–266. Oct 8-12, Galway, Ireland.

[9] Poveda-Villalón, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.. Validating ontologies with OOPS! In: 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12); vol. 7603 of *LNAI*. Springer; 2012, p. 267–281. Oct 8-12, Galway, Ireland.

[10] Cannataro, M., Comito, C.. A data mining ontology for grid programming. In: Proceedings of 1st International Workshop on Semantics in Peer-to-Peer and Grid Computing. 2003, p. 113–134.

[11] Brezany, P., Janciak, I., Tjoa, A.M.. Ontology-based construction of grid data mining workflows. In: Data Mining with Ontologies. Hershey; 2007, p. 182–210.

[12] Diamantini, C., Potena, D., Storti, E.. Supporting users in KDD processes design: a semantic similarity matching approach. In: Proceedings of the Planning to Learn Works. 2010, p. 27–34–134.

[13] Záková, M., Kremen, P., Zelezný, F., Lavrac, N.. Automating knowledge discovery workflow composition through ontology-based planning. IEEE Transactions on Automation Science & Engineering 2011;8(2):253–264.

[14] Kietz, J., Serban, F., Bernstein, A., Fischer, S.. Data mining workflow templates for intelligent discovery assistance and auto-experimentation. In: Proc of the ECML/PKDD'10 Workshop on Third Generation Data Mining (SoKD'10). 2010, p. 1–12.

[15] Panov, P., Dzeroski, S., Soldatova, L.N.. OntoDM: An ontology of data mining. In: ICDM Workshops. IEEE Computer Society; 2008, p. 752–760.

[16] Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.. Experiment databases - a new way to share, organize and learn from experiments. Machine Learning 2012;87(2):127–158.

[17] Keet, C.M.. The use of foundational ontologies in ontology development: an empirical assessment. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC'11); vol. 6643 of *LNCS*. Springer; 2011, p. 321–335. Heraklion, Crete, Greece, 29 May-2 June, 2011.

[18] Khan, Z., Keet, C.M.. ONSET: Automated foundational ontology selection and explanation. In: 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12); vol. 7603 of *LNAI*. Springer; 2012, p. 237–251. Oct 8-12, Galway, Ireland.

[19] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.. Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003).; 2003. Http://wonderweb.semanticweb.org.

[20] Ho, T.K., Basu, M.. Measures of geometrical complexity in classification problems. In: Data Complexity in Pattern Recognition; chap. 1. Springer; 2006, p. 3–23.

[21] Bishop, C.. Pattern Recognition and Machine Learning. Springer; 2006.

[22] Noy, N., Uschold, M., Welty, C.. Representing Classes As Property Values on the Semantic Web. 2005. W3C Working Group Note, http://www.w3.org/TR/swbp-classes-as-values/; URL `http://www.w3.org/TR/swbp-classes-as-values/`.

[23] Glimm, B., Rudolph, S., Völker, J.. Integrated metamodeling and diagnosis in OWL 2. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., et al., editors. Proceedings of the 9th International Semantic Web Conference; vol. 6496 of *LNCS*. Springer; 2010, p. 257–272.

[24] Masolo, C., Borgo, S.. Qualities in formal ontology. In: Proceedings of the Workshop on Foundational Aspects of Ontologies (FOnt 2005). 2005,Koblenz, Germany, Sept. 2005.

[25] Saunders, W., Bowers, S., O'Brien, M.. Protégé extensions for scientist-oriented modeling of observation and measurement semantics. In: Proceedings of the 6th Workshop on OWL: Experiences and Directions (OWLED 2011); vol. 796 of *CEUR-WS*. 2011,.

[26] Bowers, S., Madin, J.S., Schildhauer, M.P.. A conceptual modeling framework for expressing observational data semantics. In: Proceedings of the International Conference on Conceptual Modeling (ER'06); vol. 5231 of *LNCS*. Springer; 2008, p. 41–54.

[27] Hodgson, R., Keller, P.J.. QUDT - quantities, units, dimensions and data types in OWL and XML. Online; 2011. Http://www.qudt.org/.

[28] Serban, F., Vanschoren, J., Kietz, J.U., Bernstein, A.. A survey of intelligent assistants for data analysis. ACM Comput Surv 2013;45(3):31:1–31:35.

[29] Nguyen, P., Kalousis, A., Hilario, M.. A meta-mining infrastructure to support KD workflow optimization. In: Proc of the ECML/PKDD11 Workshop on Planning to Learn and Service-Oriented Knowledge Discovery. 2011,.

[30] Ławrynowicz, A., Potoniec, J.. Pattern based feature construction in semantic data mining. Int J Semantic Web Inf Syst 2014;:(accepted).

[31] Nguyen, P., Kalousis, A., Hilario, M.. Experimental evaluation of the e-LICO meta-miner. In: Proceedings of the International Workshop on Planning to Learn (PlanLearn 2012); vol. 950 of *CEUR Workshop Proceedings*. CEUR-WS.org; 2012,.